# EMBEDDED LINUX SYSTEMS

Course Code: ECE340
Semester: V

**COURSE OBJECTIVES:**
This course enables the learner to:
1. Acquire knowledge on Embedded Linux architecture, kernel internals, and kernel customization procedures.
2. Develop device driver programming, hardware interfacing, and user-space applications using Embedded Linux.

**UNIT-I (11 periods)**
**Introduction to Embedded Linux and Toolchain**
Introduction to Embedded Systems, Embedded Linux vs. Desktop Linux, Monolithic vs. Microkernel-Linux Architecture (Kernel, Userspace, System Libraries)-Bootloader (ROM Code, Secondary Program Loader)-Kernel (Decompression, Setup, Startup)-Init Process (systemd, SysVinit)-Essential Shell Commands (File, Text Processing, Networking)-Shell Scripting (Variables, Conditionals, Loops, Functions) Cross-compilation concept and setup (arm-linux-gnueabihf-), using GCC, GNU Make for build automation, and debugging with GDB (local and remote).

**UNIT-II (12 periods)**
**Kernel, Root Filesystem, and Hardware Interfacing**
Linux Kernel and Root Filesystem-Kernel configuration and compilation-Kernel modules and loading-Root filesystem layout and BusyBox-Introduction to Device Drivers-Character device drivers-Device tree basics-GPIO, UART, I2C interfacing.

**UNIT-III (11 periods)**
**Build Systems and Bootloaders**
Embedded Linux Build Systems and Bootloaders-Overview of Yocto and Buildroot-Bootloader basics: Configuration, Compilation, Porting to a new board, U-Boot Environment Variables, Boot Commands, and interacting with hardware-Flashing embedded Linux images.

**UNIT-IV (11 periods)**

**Real-Time Application Development**
Concepts - User-space application programming - Process control, multi-threading - Introduction to real-time constraints, the PREEMPT_RT patch, configuring a preemptible kernel, and measuring latency (cyclictest) - Debugging tools and performance monitoring.

**TEXT BOOKS**
1. Streif, Rudolf J. Embedded Linux Systems with the Yocto Project, Prentice Hall Press, 2016.
2. Simmons, Chris. Mastering embedded Linux programming. Packt Publishing Ltd, 2015.
3. Corbet, Jonathan, Alessandro Rubini, and Greg Kroah-Hartman. Linux device drivers. O'Reilly Media, Inc., 2005.

**REFERENCES**
1. Blum, Richard. Linux command line and shell scripting bible. Vol. 481. John Wiley & Sons, 2008.
2. Langbridge, James A. Professional Embedded ARM Development, John Wiley & Sons, 2013.
3. Yaghmour, Karim. Building embedded Linux systems, O'Reilly Japan, 2003.

**ONLINE MATERIALS**
1. NPTEL - https://nptel.ac.in/courses/117106113
2. NPTEL - https://elearn.nptel.ac.in/shop/nptel/embedded-linux-batch-2/
3. NPTEL - https://elearn.nptel.ac.in/shop/partnering-courses/registration-ongoing/embedded-linux-batch-v/

**UNIT-WISE LEARNING OUTCOMES**
**Unit I:**
• Explain the architecture and boot process of Embedded Linux systems and apply essential Linux commands/shell scripting to develop and test embedded applications.

**Unit II:**
• Develop basic device drivers for peripheral interfacing to design a customized Embedded Linux system.

**Unit III:**
• Develop and deploy Embedded Linux systems by configuring and building custom images using Yocto or Buildroot, porting and customizing bootloaders (U-Boot), and flashing images onto target hardware.

**Unit IV:**

• Develop and analyze user-space applications with process control and multithreading, while configuring and evaluating real-time performance of Embedded Linux systems using PREEMPT_RT and debugging or performance monitoring tools.

**LIST OF LABORATORY EXERCISES**

1. Introduction to essential Linux shell commands and scripting.
2. Cross-compilation of a simple C program for the ARM platform using gcc and tool chains.
3. Building, customizing, and booting the Linux kernel on Raspberry Pi.
4. Developing and inserting a basic Linux kernel module (character device driver).
5. Accessing and controlling GPIO pins from Linux user space.
6. Interfacing sensors using I2C protocol and performing data read/write operations.
7. Cross-compiling and running a Python-based IoT application using GPIO and I2C sensors.
8. Creating and managing system services for automation on startup.